Методичка 2.2 - Регистры процессора

Семейство процессоров x86

Лидерами рынка в семействе процессоров с архитектурой x86 являются компании Intel и AMD. Семейство процессоров x86 включает в себя архитектуры x86 и x64. Сейчас практически все процессоры на современных компьютерах поддерживают архитектуру x64, она является обратно совместимой с архитектурой x86. Архитектура x64 является более современной, но с точки зрения реверс-инжиниринга разница не существенная. В рамках этого курса будут рассмотрены обе архитектуры. Сначала будет знакомство с архитектурой x86, а потом будут рассмотрены нововведения и отличиях в архитектуре x64.

Регистры процессора x86

Регистры процессора - это сверхбыстрая память внутри процессора. Ячейки этой памяти называются регистрами.

Регистры можно разделить на несколько типов по их назначению:

- Регистры общего назначения. Они могут быть использованы программистом по своему усмотрению. В них можно хранить любые данные: числа, адреса и другую информацию.
- Регистры - указатели стека. Они используются при работе со стеком.
- Индексные регистры. Используются в операциях со строками.
- Указатель команд. Указывает на инструкцию, которая будет выполнена следующей.
- Сегментные регистры. Данный тип регистров необходим для обращения к определенным сегментам памяти.
- Регистр признаков (или регистр флагов). Содержит признаки результата выполненной операции и флаги управления.

Регистры общего назначения

Регистры общего назначения можно условно разделить на 4 типа:

Аккумулятор - это регистр EAX.
База - это регистр EBX
Счетчик - это регистр ECX
Регистр данных - EDX

Данные регистры имеют размер 32 бита. Для того, чтобы можно было удобно обращаться к определенной области памяти в рамках регистра, существует регистр AX - это младшие 16 бит от регистра EAX, а также регистры AL и AH, который занимают по 8 бит в младших и старших битах соответственно. Дробление регистров EBX, ECX и EDX осуществляется аналогично регистру EAX.

В регистры общего назначения можно записать максимум 4 байта, таким образом в регистре можно хранить числа от 0 до 2^32, что равно чуть более 4 млрд. В шестнадцатеричной системе максимальное число будет выглядеть так - 0xFFFFFFFF.

Регистры - указатели стека

К регистрам - указателям относятся регистры ESP и EBP. Данные регистры позволяют работать со стеком. Регистр ESP всегда указывает на вершину стека, а регистр EBP указывает на начало стекового фрейма. Эти регистры имеют регистры для обращения к младшим битам. Это регистры SP и BP.

Индексные регистры

Регистры - указатели ESI и EDI используются, например, при копировании строк из одного буфера в другой. Регистр ESI называется индексом источника, а регистр EDI индексом приемника. Они, также как и регистры общего назначения, имеют регистры для обращения к младшим битам регистра. Это регистры SI и DI.

Регистры состояния и управления

Регистры состояния и управления постоянно содержат информацию о состоянии как самого процессора, так и программы, которая в данный момент выполняется. Данная группа содержит 2 регистра:

Регистр признаков (или регистр флагов) - это EFLAGS
Регистр указателя команды - это EIP

Данные регистры имеют размер - 32 бита. Также есть регистры FLAGS и IP, которые позволяют обратиться к младшим битам. Используя эти регистры, можно получать информацию о результатах выполнения команд и влиять на состояние процессора.

Сегментные регистры

Сегментные регистры используются для обращения к тому или иному сегменту памяти. Данные регистры можно разделить на 4 типа:

Регистр кода - это CS
Регистр данных - это DS
Регистр стека - это SS
Дополнительные регистры - это ES, FS и GS

Все сегментные регистры имеют размер 16 бит. Запись чисел напрямую в данные регистры невозможна, только через другие регистры, например, через регистры общего назначения.

Базовые операции с регистрами процессора

Первый оператор, с которым необходимо познакомимся, это оператор MOV. Данный оператор позволяет загружать значение в регистр. Например, инструкция

mov eax, 4

загружает значение 4 в регистр EAX.

Следующие операторы это - ADD, SUB, INC и DEC. Операторы ADD и SUB позволяют выполнять сложение и вычитание. Например, инструкция

add ebx, 2

прибавляет значение 2 к значению, которое было записано в регистре EBX.

Операторы INC и DEC позволяют увеличить число на единицу или уменьшить его ровно на единицу. Это операции инкремент и декремент. Например, инструкция

inc ecx

Увеличивает значение в регистре ECX на единицу.

Оператор XOR - это сложение по модулю 2 или исключающее или. Операция XOR очень часто используется компиляторами для обнуления значения регистра, так как XOR числа с самим собой будет давать 0. Например, инструкция

xor edx, edx

загрузит значение 0 в регистр EDX независимо от того, какое значения было в регистре перед операцией.

xchg eax, ebx

оператора XCHG меняет значения местами в регистрах.

Также можно обращаться не только как к значению в регистре, но и как к адресу, для этого достаточно регистр записать в квадратных скобках.

mov eax, [ebx]

Это означает загрузить в регистр EAX значение, которое лежит по адресу, который указан в регистре EBX. Это важно понимать, так как компиляторы часто используют эту возможность.

Еще есть оператор LEA, который тоже часто встречается.

lea eax, [ebx + 5]

В отличие от оператора MOV, оператор LEA копирует не значение по адресу, а значение после арифметической операции в квадратных скобках. Таким образом, он к значению, которое хранится в EBX прибавит 5 и запишет результат в EAX.

Разработка программы

Исходный код программы prog-1.2.asm

```
.386
.model flat, stdcall

_text segment
start:
xor eax, eax
add eax, 4
sub eax, 2
inc eax
inc eax
inc eax
mov ebx, 0
            add bx, 8
            dec bx
            xor ebx, ebx
            xchg eax, ebx
            lea eax, [ebx + 5]
            mov eax, 00402000h
            mov ecx, [eax]
            ret
_text ends
end start
```

Трансляция:

> ml /c /coff prog-1.2.asm

Линковка:

> link /SUBSYSTEM:CONSOLE /DYNAMICBASE:NO prog-1.2.obj

При линковке использовался параметр /DYNAMICBASE:NO. Этот параметр отключает рандомизацию адресного пространства для того, чтобы адрес загрузки программы был всегда одинаковый. По умолчанию в ОС Windows все программы загружаются в память по адресу 0x00400000.

Запуск.

> prog-1.2.exe

Программа что-то сделала и завершилась. Для того, чтобы увидеть, как в процессе ее выполнения изменяются значения в регистрах, необходимо запустить программу под отладчиком.

Для отладки будет использоваться отладчик OllyDbg.

Рекомендации по установке отладчика OllyDbg

Ссылка: http://www.ollydbg.de/odbg110.zip

Распаковывать архив в "C:\Program Files (x86)\odbg110" и создать ярлык на рабочем столе.

Если при первом запуске отладчик предложит заменить некоторые библиотеки на более современные, то можно отказаться.

Также можно отключить следующее предупреждения:

Options - Debugging Options - Security

Warn when terminating active process
Warn if not administrator
Warn when breakpoint is outside the code section

Подобрать удобный шрифт и цвета:
Appearance - Font (all) - <здесь ваш выбор>
Appearance - Colors (all) - <здесь ваш выбор>

Запуск отладчика.

На данном этапе необходимо запомнить несколько команд:

F3 - открыть файл

F2 - создать точку останова
F8 - выполнить следующую инструкцию
F9 - продолжить выполнение программы
Ctrl + F2 - перезапустить программу
Ctrl + G - перейти по адресу

Открываем программу prog-1.2.exe (F3).

Переходим к секции кода (Ctrl + G, 0x00401000). В следующем уроке вы узнаете, почему мы выбрали именно этот адрес.

Ставим точку останова (F2).

Продолжаем выполнение программы (F9).

Выполняем пошагово программу (F8) и наблюдаем за изменением значений в регистрах.

Для того, чтобы начать выполнение программы сначала необходимо перезапустить программу (Ctrl + F2).